STAT 511-611

SAS Project

Deadline: Oct 21st, 2019

The project is open-book take-home. You *may not ask questions from others, work with others, or obtain answers from others* on the project. *work alone* on this project. If a question is not clear, ask me, not another student. If you can’t figure out an answer, think about it a bit more—but do *not* ask someone else for ideas.

How to submit project files:

1. A file of your SAS code. Name this file: SASProject\_Firstname\_Lastname\_Username.sas. For me it will be:

SASProject\_Minh\_Pham\_mtpsma.sas.

In this file the problems must be listed in order. Each problem must also have a comment line that precedes it. For example, for question 2, use:

\*\*\* q2 \*\*\*;

Each *part* of a problem must also have a comment line that precedes it. For example for question 2b, use:

\*\*\* q2b \*\*\*;

This ensures that I and the graders can follow your work.

1. A file of your SAS output (both PROC’s and the SAS LOG).

I suggest saving the output and the log file in a RTF file. This can be done by:

Run your code then go to RESULTS tab. Click on Download results as a RTF file.

Then you can click on the LOG tab and copy the LOG of the run.

Save both the log and the output in a RTF file and name it:

SASProjectLog\_Firstname\_Lastname\_Username.rtf

Also:

* + 1. Each problem that requests specific output from the SAS log *must* begin and end with a %put statement that lists the question ID. For example, if I request that some output from a DATA step to the SAS log for q1b, use the following in the your SAS code:

%put q1b;

DATA new1;

…

run;

%put q1b;

* + 1. Each problem that uses one or more PROC’s must have a TITLE statement and a FOOTNOTE statement that contains the correct question reference number and letter. As a result, for example, for question 1a the output file would then contain

q1a

(output from the PROC)

q1a

For example: For question 1 part a

Title “q1a”;

Data mydata;

INFILE myfillocation;

INPUT myvar1 myvar 2;

Run;

Footnote “q1a”

These should be the only occurrences of “q1a” in the output file. For example, do not give variable or SAS data sets names such as “q1a” or “q1aData”.

* + 1. This output file should not be modified in any way by you. (If we run your code from project sas file (with possibly new directory-name definitions), and if you were able to create all of the SAS data sets on your own we should obtain a file that is *identical* to yours.)

You can use this same guideline to create output for your assignments.

Some questions may require you to learn some features that you have not yet seen. (This is what will happen in many cases when you work on real problems.) In these cases (and in possibly other cases), some hints are included at the end of the questions. If you decide you want to look at them, please do. This does not mean they will always help you! You may have figured out a different way (maybe even a better way) to solve the problem than what my hints suggest. Some earlier hints may be useful later; some later hints may be useful earlier.

You must write code to handle all the cases that I ask you to handle, even if they do not occur in the particular data set that you are analyzing. For example, suppose a question says “If the first character in the variable NAME is ‘Z,’ then print “The name starts with Z”. Then you must write code to handle this situation even if none of the data has a name that starts with Z.

The code should be general enough that it can be used on other, similar, data sets. For example, suppose I ask you to do a check on each line for a certain error and to print out the variable X if you find that error. If you look at the data and you see that such errors are in lines 6 and 8, do not write code like this: “If \_N\_=6 or \_N\_=8 then print X;” !

If you do not understand what a question is asking, please email me directly at mtpsma@rit.edu. Do *not* submit such questions (or any postings) about this project to the Discussion Groups in myCourses.

Data files: *Pr1Snowfall1.csv*, *Pr1Snowfall2.csv, Pr1CT.dat*

**Problem 1 (15pts, Bonus 2pts):**

In Assignment 2, you were asked to read in “the data for snowfall *from the 1884-85 season until the 2001-02 season* (the 2002-03 season is incomplete so please eliminate it from consideration).” The SAS solution set included this line of code:

INFILE "&dirdata.RochesterSnowfall.csv"

DLM=',' FIRSTOBS = **5** OBS = **122** ;

The problem with this coding is that you manually needed to look in the file to find that you had to start on line 5 and end on line 122. For this reason, and because your supervisor has recently told you that many such files will need to be processed, it is not reasonable to have to look in each file manually to find the starting and ending lines. Instead, you want *write code* to read in the correct information from such files. To be more specific, here is what you know about the file structure. (You can guess from *RochesterSnowfall.csv* that this is the structure, but it is being supplied here for clarity):

1. The files are comma-delimited (like *RochesterSnowfall.csv*).
2. The lines that contain the snowfall information are of the form shown in *RochesterSnowfall.csv*. Examples:

1884-85,0,T,1,27.1,22.2,17,3.5,19.5,T,90.3,

2001-02,0,T,0.1,7.1,11.9,18.7,13.8,6.5,T,58.1,

2002-03,0.0,0,16.9,41.1,43.4,21.9,,,,,

1. In particular, the lines that contain the snowfall information always begin with “xxxx” in columns 1-4, where xxxx is a 4-digit number. (The numbers should also be between 1800 and 2200, so the code can be used for a long time. However, you do not need to check this 1800-2200 range in your code.)
2. These snowfall-information lines occur sequentially in the data file.
3. The lines that do not contain the snowfall information never begin with such an “xxxx”.
4. The lines that contain the snowfall information always begin with xxxx-yy in columns 1-6, where xxxx-yy is the Season variable.
5. The remaining parts of the line that contain the snowfall information are either numeric or “T” (with one more exception that is noted in the next paragraph).
6. The last line of snowfall information may be incomplete. Such an incomplete line has a missing value (blank entry) for the Total variable (as well as at least one missing value for the months). All other lines of snowfall information will be complete.

Also:

1. Only the variables   
    Year,Season, Sep,Oct,Nov,Dec,Jan,Feb,Mar,Apr,May,Total  
   should be included in the output data set, *and in this order*. Here, Year is numeric and is equal to the “xxxx” above, Season is equal to the “xxxx-yy” above, and Sep--Total is numeric.
2. Your code should convert each “T” to a 0.
3. Only full seasons should be included. (In the above example, the 2001-02 season is a full season—all of the values have been entered; the 2002-03 season is not a full season—some of the values have not been entered. This is mentioned in item 8 above.)

With this information:

1. (10 pts q1a) In SAS, write code to read in such a file. Use *Pr1Snowfall1.csv* as your data file—do ***not*** use *RochesterSnowfall.csv* for this problem. Your code must work not just for this particular file, of course, but for other files that have this same format. Name your SAS data set *snow1*. Use PROC CONTENTS;RUN; and PROC PRINT;RUN; to show your results.
2. Bonus (2pts q1b) After you have run q1a, print out

N full years of snowfall data

on the SAS log, where N is the number of full years of data and N is obtained from code. For example, if there are 101 full years of snowfall data, the log file should include

1. ll years of snowfall data
2. (5pts q1c) Do the same work as in (a), but for *Pr1Snowfall2.csv*. Name this SAS data set *snow2* and run PROC CONTENTS and PROC PRINT again. Your code to do this should be *exactly* the same code in (a) and (b) except for the data file name and the SAS data set name.

**Problem 2 (25pts):**

You are working with a group of biostatisticians on a large clinical trial. You have been asked to read in and perform a series of checks on some of the data for this trial, and to find some basic summary measures.

*It may be a good idea to read all of these questions first before you do any work.*

The data are in the tab-delimited file *Pr1CT.dat*. This data file contains:

* PatientID: format is Xyyy, where X is a code (values should P, J, M, N, or V) that indicates where the hospital is located and yyy is a 3-digit number whose values should be integers between 101 and 999. However, the text “Missing” is also a possible value for PatientID.
* Sex: should be M, F, or “Missing”.
* Race (or Ethnicity): should be Caucasian, Asian, Black, Hispanic/Latino, Other, or “Missing”.
* Treatment: T (treatment group) or C (control groups) for a potential anti-allergen drug. (The drug is intended to be taken once each week, but that does not matter for this problem.) There should not be any value of “Missing” – but there may be.
* Sens0: An allergen-sensitivity measure to the drug at “time 0,” just before the treatment or control was given to the patient for the first time. Lower numbers are better.
* Sens1, Sens2, Sens6, Sens12, Sens24: the same measurement at 1, 2,…, 24 months after time 0.

In some of the questions below, you will be asked to check some parts of the data for errors. There may be other errors in the data set as well, such as misspellings. In real life, you would also want to detect those errors. However, for this project, please ignore those other errors.

1. (5pts q2a) Read the data into the SAS data set CT1. Name the variables as indicated in the data file or as shown above. *Make sure that the variables* *are in this same order in the SAS data set*. Write code to change any “Missing” values for *all* of the character variables into the appropriate SAS missing value code. (If you are unable to create this SAS data set, which you will need for later questions, you may send me an email to request it. Your score for this question will then be 0/5, and there will be no need to run the next PROC CONTENTS or PROC PRINT.) Then run PROC CONTENTS data=CT1; RUN;   
   PROC PRINT data=CT1 (OBS=325);RUN;
2. (5pts q2b) Using CT1 as an input SAS data set, split the PatientID values into Location (first digit) and IDNumber (next 3 digits), and write the resulting updated SAS data set—that is, with these two additional variables—to CT2. The Location and IDNumber variables should become the last two variables in the SAS data set. Both Location and IDNumber should be character variables.

You may assume for this Project that PatientID always has exactly 4 characters (Xyyy) unless its value is “Missing”. (For a real problem, this should be checked.)

(If you are unable to create this SAS data set (which you will need for later questions), you may send me an email to request it. Your score will then be 0/5, and there will be no need to run the next PROC CONTENTS or PROC PRINT.) Then run   
PROC CONTENTS data=CT2; RUN;   
PROC PRINT data=CT2 (OBS=325);RUN;

1. (5pts q3c) Using CT2 as the input SAS data set, use a DATA step to verify that all non-missing Location values are P, J, M, N, or V. If they are, print

All Location values are correct

to the SAS log file. The code is expected to work as requested whether or not all Location values are correct. That is, you need to include the correct logic in the code, whether or not all all Location values are correct.

If not, then for each incorrect value, print a line that looks like this. (This line is for PatientID="Q124" but of course you need to use the actual incorrect value.) Also, change the incorrect location value to “X” (but do *not* make any changes to the Patient ID value).

Error: Location Q changed to X for PatientID Q124.

Write the resulting file to the SAS data set CT3. Then run   
PROC PRINT data=CT3 (OBS=325);RUN;

1. (5pts q3d) Using CT3, check whether all non-missing IDNumber values are, in fact, integers, and are between 101 and 999. If so, then print  
    All IDNumber values are correct

to the SAS log. If not, then for each incorrect value, print a line that looks like this. (This line is for PatientID="V094" but of course you need to use the actual incorrect value.) Also, change the incorrect IDNumber value to a 999 (but do not make any changes to the Patient ID value).

Error: IDNumber 094 changed to 999 for PatientID V094.

Write the resulting file to the SAS data set CT4. Then run   
PROC PRINT data=CT4 (OBS=325);RUN;

1. (5pts q3e) Using CT4, use PROC TABULATE to find means and standard deviations of Sens0, Sens1,… Sens24 for each Treatment group. Put treatment groups (C and T) and the Mean and Std within each treatment group in the rows, and Sens0, Sens1,… Sens24 in the columns (so, you should have a 4×6 table of the sensitivity summaries). Round your answers in the table to the nearest integer. If you can’t achieve this 4×6 format, use another one (-1 or -2).

**Problem 3 (30pts):**

Monthly weather information has been obtained for weather stations in four cities in upstate New York: Buffalo, Rochester, Syracuse and Albany. This information is contained in three .csv files, whose names should make the contents obvious: *Weather\_Buffalo.csv, Weather\_Rochester.csv,* and *Weather\_SyracuseAlbany.csv.*

Problem 3, 4, and 5 deal with these 3 data sets.

The weather information contains monthly readings that start at 1900/01/01 (or later, depending on when the weather station started collecting data) and end at 2013/09/01 (or earlier, depending on when the weather station stopped collecting data). In some cities, more than one weather station has collected the data.

The first five monthly readings for a weather station in Buffalo look like this. The readings are transposed here to make them easier to read—in the data file, these columns are rows and these rows are columns. I also added column (really, row, or variable) numbers to this example.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | STATION | GHCND: USC00301010 | GHCND: USC00301010 | GHCND: USC00301010 | GHCND: USC00301010 |
| 2 | STATION\_NAME | BUFFALO NY US | BUFFALO NY US | BUFFALO NY US | BUFFALO NY US |
| 3 | ELEVATION | 234.1 | 234.1 | 234.1 | 234.1 |
| 4 | LATITUDE | 42.88333 | 42.88333 | 42.88333 | 42.88333 |
| 5 | LONGITUDE | -78.88333 | -78.88333 | -78.88333 | -78.88333 |
| 6 | DATE | 19000101 | 19000201 | 19000301 | 19000401 |
| 7 | MXSD | 102 | 203 | 635 | 0 |
| 8 | Missing | 0 | 0 | 0 | 0 |
| 9 | Consecutive Missing | 0 | 0 | 0 | 0 |
| 10 | TPCP | 972 | 1327 | 965 | 288 |
| 11 | Missing | 0 | 0 | 0 | 0 |
| 12 | Consecutive Missing | 0 | 0 | 0 | 0 |
| 13 | TSNW | 241 | 649 | 854 | 13 |
| 14 | Missing | 0 | 0 | 0 | 0 |
| 15 | Consecutive Missing | 0 | 0 | 0 | 0 |
| 16 | MMXT | 9 | -12 | -1 | 103 |
| 17 | Missing | 0 | 0 | 0 | 0 |
| 18 | Consecutive Missing | 0 | 0 | 0 | 0 |
| 19 | MMNT | -57 | -83 | -72 | 25 |
| 20 | Missing | 0 | 0 | 0 | 0 |
| 21 | Consecutive Missing | 0 | 0 | 0 | 0 |
| 22 | MNTM | -24 | -47 | -37 | 64 |
| 23 | Missing | 0 | 0 | 0 | 0 |
| 24 | Consecutive Missing | 0 | 0 | 0 | 0 |

Here is the meaning for each variable. Note that there are variables named *Missing* and *Consecutive Missing* for each of the six *measured variables* (the snow and precipitation variables)—the meaning, however, is only given here for the first pair.

|  |  |  |
| --- | --- | --- |
| **Variable** | **Meaning** | **Example** |
| STATION | Station identification code | GHCND: USC00301010 |
| STATION\_NAME | Usually, a city or airport name | BUFFALO NY US |
| ELEVATION | Above mean sea level (thousandths of meters) | 234.1 |
| LATITUDE | Latitude | 42.88333 |
| LONGITUDE | Longitude | -78.88333 |
| DATE | (Monthly) Year 4 digits; month 2 digits; day 2 digits | 19000101 |
| MXSD | Maximum snow depth during the month (mm) | 102 |
| Missing | N of days MSXD is missing in that month | 0 |
| Consecutive Missing | Maximum N of consecutive days in that month that MXSD is missing. | 0 |
| TPCP | Total precipitation for month (tenths of mm) | 972 |
| Missing |  | 0 |
| Consecutive Missing |  | 0 |
| TSNW | Total snow fall for month (mm) | 241 |
| Missing |  | 0 |
| Consecutive Missing |  | 0 |
| MMXT | Monthly mean maximum temperature (tenths of °C) | 9 |
| Missing |  | 0 |
| Consecutive Missing |  | 0 |
| MMNT | Monthly mean minimum temperature (tenths of °C) | -57 |
| Missing |  | 0 |
| Consecutive Missing |  | 0 |
| MNTM | Monthly mean temperature (tenths of °C) | -24 |
| Missing |  | 0 |
| Consecutive Missing |  | 0 |

For this project, you will eventually need to rename some of these variables, as follows:

Variable New Name

STATION\_NAME Station

ELEVATION Elevation

LATITUDE Latitude

LONGITUDE Longitude

DATE Date

MXSD MaxSnow

TPCP Precip

TSNW Snowfall

MMXT MeanMaxTemp

MMNT MeanMinTemp

MNTM MeanTemp

1. (5pts q3a) Read in the Buffalo csv file using a DATA step. Because SAS does not read in the names, use the following names *and put them in this order in the SAS data set*:

Station\_ID Station Elevation Latitude Longitude Date MaxSnow Miss CMiss Precip Miss\_1 CMiss\_1 Snowfall Miss\_2 CMiss\_2 MeanMaxTemp Miss\_3 CMiss\_3 MeanMinTemp Miss\_4 CMiss\_4 MeanTemp Miss\_5 CMiss\_5

Convert all -9999’s (you only need to check this for the 6 measured variables) to the SAS missing value code. Read in the date field correctly, and use the format YYMMDD10. to display it.. Name this data set *wBuff1*.

Output: (1) a PROC CONTENTS of this data set and (2) a PROC PRINT of the first 20 records. (Make sure the output width is wide enough (up to the max width of ls=256))

1. (5pt q3b) Find the total number of missing days for each of the Miss and CMiss columns (12 columns in total) and display these in an appropriate listing. The listing should include each variable name and the total number of missing days.

Output: the listing. (Again, make sure the output width is wide enough.)

1. (5pts q3c) Drop these Miss and CMiss columns (12 columns in total) from your data set, and also drop the Station\_ID, Elevation, Latitude, and Longitude columns. Name this data set *wBuff2*. Then, using code, find how many unique Station names exist in the data set.

Output: (1) a PROC CONTENTS of this data set and (2) a listing of the unique Station names.

1. (5 pts q3d) Rename the stations from their long all-capital-letter versions to “Buffalo Airport” and “Buffalo City”. Also create two new character variables (and in this order in the SAS data set, and after the other variables): City and Site. For this problem, City would be “Buffalo” and Site would be either “Airport” or “City”. Name this data set *wBuff3*.

Output: (1) a PROC CONTENTS of this data set and (2) a PROC PRINT of the first 20 records.

1. (5pts q3e) Add more date-based variables, as follows (and in this order in the SAS data set, and after the other variables):
   * 1. MonthN, the month number (1, 2, …, 12) for the date.
     2. Month, the abbreviated month name (“Jan”, “Feb”, … , “Dec”) for the date.
     3. Year, the year of the date.
     4. SnowSeasonLong, the year of the longer version of the snow season. Here is an example: for month numbers 10, 11, 12 of 1930 and month numbers 1, 2, 3 ,4 of 1931, set SnowSeasonLong to 1930. For any other month numbers, set SnowSeasonLong to be the SAS missing value code. (Do this for all years, of course.)

Name this data set *wBuff4*.

Output: (1) a PROC CONTENTS of this data set and (2) a PROC PRINT of the first 20 records.

1. (5pts q3f) Clean up the measured variables, as follows. Convert the 3 snow and precipitation values into inches (to the nearest 0.1), Convert the 3 temperature values into °F (to the nearest 0.1). *Make sure you do these correctly*. Name this data set *wBuff5*.

Output: a PROC PRINT of the first 20 records.

**Problem 4 (15pts):**

1. (10pts q4a) Verify the extent of any overlapping data between the Airport and City sites.

Output: a (SAS data set) listing (PROC PRINT) of every month for which the Airport and City sites have overlapping months (dates) (but not necessarily overlapping useful data—because of any original -9999 coding, some of the records may have missing data).

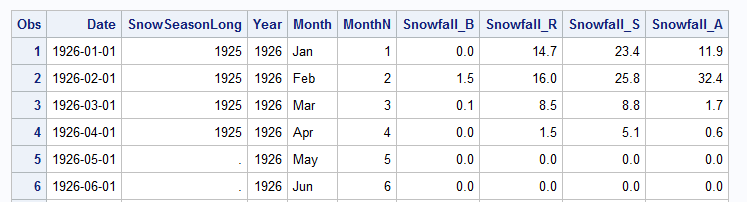
For this particular answer, your listing should include only five variables, and in this order (the SAS data set itself does not need to have the variables in this order): Date of overlap, Snowfall amount at Airport, Precip at Airport, Snowfall amount at City, Precip at City. For variable names, use Date,Snowfall\_Air,Precip\_Air,Snowfall\_City,Precip\_City.

1. (5pts q4b) Regardless of your previous answer, use 1943-07-01 as the date to stop using the City data and begin using the Airport data. Create a new SAS data set from wBuff5 that only contains this subset of rows. (So, this new data frame should still have the same starting and ending dates, but with no more overlapping months.) Name this SAS data set wBuff.

Output: a PROC CONTENTS of this SAS data set.

**Problem 5 (20pts):**

Similar work has been done for you for Rochester, Syracuse, and Albany. These are available in the three files of permanent SAS data sets *wRoch.sas7bdat, wSyr.sas7bdat, and wAlb.sas7bdat*. You will now need to merge all 4 cities together (from the corresponding 4 SAS data sets, of course), and then create summaries. We are only going to look at the Snowfall data for the rest of the questions in this project, so that is the only measured variable you should keep here. Also, only keep the dates that *all 4 cities have in common*. Your final SAS data set (if you answer the first question below correctly) should then look like this for the first 6 records:



* 1. (10pts q5a) Merge all four SAS data sets. Please merge (from left to right) wBuff, wRoch, wSyr, and wAlb. I want the resulting SAS data set to contain Date, the other date-based variables (see above listing), Snowfall\_B (the Snowfall values for Buffalo), and similarly Snowfall\_R, Snowfall\_S and Snowfall\_A. Name this SAS data set wAll. If you did this correctly, you should get the listing above for the first 6 records.

Output: PROC CONTENTS for wAll; PROC PRINT for wAll, first 20 records. Make sure the PROC PRINT listing lists the data in the order shown above (but it does not need to be in this order in the SAS data set itself).

* 1. (10pts q5b) (for 611-01 and 611-02 students only) Summarize the data. First, find the yearly total snowfall during the long snow season (from October to April) for each city. Exclude the 1925 season because it is incomplete. Save your results in a SAS data set that has the variables SnowSeasonLong, Snowfall\_B, Snowfall\_R, Snowfall\_S, and Snowfall\_A. You should, of course, have one row for each season. Name this SAS data set wAllYearSumsSL. Using this data set, find the overall mean, standard deviation, and %CV for the snowfall in each of the four cities. Round the results to the nearest inch (or nearest %), and put your results in a table that is formatted like this:

Snowfall\_B Snowfall\_R Snowfall\_S Snowfall\_A

mean xx xx xx xx

std xx xx xx xx

CV xx xx xx xx

Output: (1) PROC CONTENTS for wAllYearSumsSL; (2) PROC PRINT for wAllYearSumsSL, first 20 records (with variable listing in this order: SnowSeasonLong, Snowfall\_B, Snowfall\_R, Snowfall\_S, Snowfall\_A); and (3) the mean/std/CV table.

**Problem 6 (16pts)**

Consider a data set that contains the variables x1-x500 in a space-delimited file *Pr3x1x500.txt*. Please begin by reading this data set into the SAS data set x1x50, with var names x1-x500, but only keep x1-x50 and only keep the first 10 records of the 100 in the original data set. This SAS data set will be used for testing purposes.  
 The problem is that you have been told that these 50 variables are actually in groups of 5, and should have been labeled with the prefixes a,b,c,d,e. For example, x1-x5 🡪 a1-e1, x6-x10🡪a2-e2, and so on. This happens often, so you have been told to write a SAS macro to handle not just this situation, but the following more general one:

* A SAS data set contains a sequence of variables—for simplicity, let’s assume these are the only variables in the data set. Examples would be x1-x50 or test1-test300. The sequence always starts at 1 but the end (e.g., 50 or 300) can change from one situation to another, and the prefix (e.g., “x” or “test”) can also change.
* These variables need to be re-labeled into groups of N, where N (e.g., 5) can change. The group prefixes (e.g., “a”, “b”, “c”, “d”, “e”) can also change.
* The variables in the SAS output data set should be arranged in prefix order. In our example, this would be a1-a10, b1-b10, …, e1-e10.
* If the end of the original sequence (e.g., 50) does not divide exactly into the number of groups (e.g., if there were 6 groups instead of 5 for the set of 50) then the macro should print out a descriptive error that must start with the word “Error” and must include the values that caused the problem (e.g., “50” and “6”). After this print out, the macro should stop.
* Aside from this check you need to make, you may assume for purposes of this macro writing that all the arguments will be correctly entered by the user.

1. (13pts q6a) Write the macro. Use these arguments, and only these arguments, in the macro.

datain: name of i/p SAS data set

dataout: name of o/p SAS data set

seqEnd: last index of i/p sequence (in our example, 50)

prefix: prefix used in the sequence (in our example, x)

grpPrefs: list of prefixes used in the new sequences (our ex: a b c d e)

Then test it with the data set and situation given (x1-x50 to the new prefixes)

Then test it again with the data set given but with only two prefixes, pre and post.

Output: your macro definition (we will look at this in your file of SAS code); a PROC CONTENTS (*use the order=varnum option*) of the first new SAS data; a PROC PRINT of the first 5 records of this data set (but don’t worry if all the variables for a record don’t fit on one line); a similar PROC CONTENTS for the second new SAS data set; a similar PROC PRINT for the second new SAS data set.

Do *not* run option MPRINT or MLOGIC here as part of the output file for this project; do *not* have the macro print out what it is doing as it is doing its work as part of the output file for this project.

1. (3pts q6b) (For 611-01 and 611-02 students only) Test this again on the same SAS data set (with x1-x50) but this time using 11 groups (with prefixes equal to the first 11 letters of the alphabet)

Output: the SAS log, including the macro call and error message.

Hints:

* Problem 1:
  + 1. If you have variables a, b, and c, and use   
     INPUT @10 b c @; … ; INPUT @1 a …;  
    the 2nd INPUT statement goes back to the beginning of the line.
  + 2. Character functions, such as SUBSTR
  + 3. The INPUT function. (Look this up in TLSB—this is *not* the same as the INPUT statement!)
  + 4. If you were the SAS program, what decisions would you need to make to read in Name correctly?
* Problem 2c:
  + For ‘All Location values are correct’ logic: Retain statement;
  + SAS Programmer’s Books (<http://support.sas.com/documentation/onlinedoc/bookshelf/94/desktop.html>)🡪Base SAS🡪Statements🡪Dictionary of SAS Statements🡪SET statements, then SET options.)
* Problem 2d:
  + INPUT function.
* Problem 3:
  + d. “if x =:” is like “where x contains”, but “if x contains” is not allowed. Or simply use “if x =”.
  + e. For month abbreviations, you can do this the long way. Alternatively, SAS does have many formats as well as a PUT function.